Max Hoac
CS 4200.E01

CS 4200 Project 1 Report

Goal: The primary goal of this implementation of the 8-puzzle is to utilize two different heuristic functions to accomplish the goal state of the puzzle. The first function is referred to as h1 uses the algorithm relating to the number of misplaced tiles, whereas the second function, h2, uses measures of the sum of the distances of the tiles in comparison to their goal positioning.

Approach: For my implementation, I planned on having my code be capable of generating a designated number of randomized puzzles as well as having the option for the user to generate their own puzzle through a user input scanner. In order to accomplish this, I created separate class files for each differing puzzle type as well as a separate class file that depicts the format of how the separate numerical nodes will connect with one another. The A* algorithm will be implemented together in a single file known where the class will possess two separate functions regarding the different heuristic functions. Altogether these components will be utilized in a working file where the user is able to choose to generate a randomized 8-puzzle or create their own 8-puzzle. When the randomized option is chosen, the code will prompt the user for the number of puzzles they want to create. Once the input is taken, both algorithms (h1 and h2) will go through the generated puzzles until all are completed. Upon completion, the depth, search cost, and time statistics will be printed out for both heuristic functions. Additionally, when asked to solve several puzzles, a table depicting the same values will be generated. If the user input option is chosen, the code will operate similarly, however it will only solve the inputted puzzle. Once again upon completion, the depth, search cost, and time statistics for the puzzle will printed out.

Analysis:

User-input puzzles

depth	total cases	h1 search cost	h1 time (ms)	h2 search cost	h2 time (ms)
4	10	8.4	9.1	8	1.8
6	1	12	5	11	1
8	9	34.44	10.89	19.33	2.22
10	1	56	12	20	3
12	9	143.11	12.89	46.11	3.22
14	3	350.33	21	67	3.33
16	7	883.14	46.29	112.14	4.43
18	2	1727	54.5	103.5	4.5
20	7	5628.57	142.71	391.14	12.29

Max Hoac
CS 4200.E01

100 Random Puzzles

depth	total cases	h1 search cost	h1 time (ms)	h2 search cost	h2 time (ms)
10	1	81	0	27	0
12	1	135	0	43	1
13	1	305	1	95	1
15	3	669	4	119	0
16	4	955	6	215	1
17	2	1576	9	277	1
18	3	1928	10	339	1
19	6	3348	18	397	2
20	6	5430	41	575	4
21	14	8477	61	827	5
22	12	13292	98	1045	7
23	12	23021	163	1755	11
24	6	28559	197	1818	10
25	10	49467	405	2621	18
26	13	57748	459	4574	32
27	2	94635	738	6411	41
28	3	113033	979	6733	48
30	1	181292	1614	25302	177

Based on the trials resulting from the several files of solvable puzzles and 100 randomly generated puzzles, the table shows that as the depth of the puzzle increases the total search cost and total time taken to solve the puzzle increases. In terms of performance the h1 algorithm or the A* algorithm using misplaced tiles had a significantly larger search cost and time spent when compared to h2, the A* algorithm using the sum of distances. One notable observation from the results is that unlike h1, the h2 algorithm only surpassed a time of over 100ms when having to solve a puzzle with a depth of 30. Additionally, the search cost resulting from the 2nd algorithm was nearly 10x lower than the search cost resulting from the 1st algorithm. Regarding the randomized puzzle data itself, the majority of cases that were recorded were depths between the values of 21 to 26.